Масштабирование CRON-задач в распределенных системах

С. В. Козлов, И. А. Карпов

Аннотация **– В статье кратко сформулировано назначение CRON-задач, как механизма запуска задач по расписанию. Приведены примеры продуктовых задач, выполняющихся по расписанию. Кратко описана необходимость горизонтального масштабирования сервисов в микросервисной архитектуре. Описаны проблемы, возникающие при горизонтальном масштабировании CRON в распределённых системах: в зависимости от количества реплик повторное или многократное выполнение логики, гонки при работе с данными, приводящие к аномалиям, дополнительная нагрузка на внешние узлы. Рассматриваемые проблемы проиллюстрированы случаями из продуктовой разработки: еженедельная рассылка электронных писем и обнуление накопленной внутренней валюты. Представлено описание и анализ пяти способов масштабирования: вынесение CRON в отдельный сервис, назначение лидера среди реплик, использование kubernetes CronJob, применение дополнительных атрибутов в базе данных и использование очередей задач на примере Bull. В четвертом способе дополнительные атрибуты в таблицах выступают в роли флага, определяющего, началось ли выполнение конкретной задачи, что позволяет избежать дублирования. Описаны нюансы реализации, преимущества и недостатки каждого из способов. Актуальность статьи обусловлена повсеместным использованием задач, выполняющихся по расписанию в высоконагруженных распределенных системах и необходимостью их надежного, предсказуемого и отказоустойчивого масштабирования.**

Ключевые слова **– CRON-задачи, масштабирование, микросервисы, отказоустойчивость, гонка данных, программная архитектура, высоконагруженные системы, консистентность данных.**

I. ВВЕДЕНИЕ

CRON – инструмент, который автоматизирует выполнение задач в запланированное время. Он работает с помощью специального файла «crontab» (сокращение от «cron table»). В этом файле в определенном формате указывается выполняемая задача и время выполнения. Например, запись вида:

*0 0 * * * команда*

означает, что указанная задача будет выполняться каждый день в полночь.

Задачи, выполняющиеся по расписанию, распространены в веб-приложениях. К ним относятся, например, очистка временных файлов, еженедельная рассылка электронных писем и генерация ежемесячных отчётов. Для автоматизации таких процессов чаще всего используется механизм CRON. В рамках монолитной архитектуры [1], которая, как правило, масштабируется вертикально с помощью наращивания вычислительных ресурсов одного сервера, его использование не вызывает затруднений. Однако в микросервисной архитектуре [2], предполагающей горизонтальное масштабирование за счет увеличения количества реплик конкретного узла для повышения его пропускной способности, возникают сложности, связанные с неоднозначным поведением CRON-задач при параллельном запуске. Одновременное выполнение одной и той же задачи на нескольких репликах приводит к ряду нежелательных эффектов: повторному или многократному выполнению логики, гонкам при работе с данными, вызывающим различные аномалии, а также к избыточной нагрузке на внешние узлы. Например, если в сервисе, развернутом в трёх репликах, реализована CRON-задача по еженедельной рассылке электронных писем, то в запланированный день каждый пользователь получит три письма вместо одного ожидаемого [3].

Выделим следующие способы решения проблем горизонтального масштабирования [4, 5] CRON-задач:

1. Вынесение подобных задач в отдельный сервис.
2. Выделение лидера среди реплик, используя механизм выбора ответственного за выполнение CRON-задач экземпляра.
3. Использование k8s CronJob.

Статья получена 24 августа 2025.

Козлов Сергей Валерьевич, Смоленский государственный университет, доцент кафедры прикладной математики и информатики, кандидат педагогических наук, доцент (email: svkozlov1981@yandex.ru)

Карпов Илья Алексеевич, Смоленский государственный университет, студент физико-математического факультета (email: ilyakarpov05@mail.ru)

4. Использование дополнительных атрибутов в базе данных, выполняющих роль флага, определяющего, началось ли выполнение задачи.

5. Использование очередей задач по типу Bull, которые аналогично CRON могут запускать job по расписанию.

В данной статье подробно рассмотрены эти способы масштабирования CRON-задач в распределенных системах. Также описаны нюансы реализации, преимущества и недостатки каждого из способов.

II. ВЫДЕЛЕНИЕ ОТДЕЛЬНОГО СЕРВИСА

Вынесение в отдельный сервис. Суть первого подхода заключается в вынесении CRON-задач в отдельный сервис, инкапсулирующий логику, связанную с планированием и запуском задач по расписанию. Такой CRON-сервис функционирует независимо от бизнес-сервисов и обеспечивает централизованное управление периодическими операциями. Важно отметить, что этот сервис должен иметь единственный экземпляр (реплику). В противном случае возможны конфликты, описанные ранее: дублирование выполнения, гонки и избыточная нагрузка. В качестве канала взаимодействия между CRON-сервисом и бизнес-сервисами могут использоваться брокеры сообщений [6] (RabbitMQ, Kafka), HTTP-запросы [7, 8] или gRPC-вызовы [9].

Рассмотрим пример из производственной практики. Допустим, в системе присутствует сервис UserService, отвечающий за управление пользователями. В условиях высокой нагрузки данный сервис масштабируется горизонтально, и в рабочем окружении поддерживается не менее трёх реплик, количество которых может динамически увеличиваться. Требуется реализовать задачу ежемесячного обнуления баллов лояльности всех пользователей. В целях предотвращения параллельного выполнения, задача запускается не в самом сервисе UserService, а в выделенном CRON-сервисе. В установленное время CRON-сервис публикует соответствующее сообщение в брокер сообщений, например, RabbitMQ, а сервис UserService, выступая в роли подписчика, обрабатывает сообщение и выполняет обнуление баллов (рис. 1).

Рис. 1. Схематичное изображение архитектуры

Такой архитектурный подход позволяет масштабировать бизнес-сервисы [10] горизонтально, создавая произвольное количество их реплик, при этом сохраняя централизованную логику запуска задач по расписанию в едином CRON-сервисе. Однако отказ этого сервиса приводит к полной остановке выполнения всех периодических задач. Кроме того, выделение отдельного узла в инфраструктуре увеличивает накладные расходы на его развертывание, мониторинг и сопровождение [11].

III. ВЫДЕЛЕНИЕ ЛИДЕРА СРЕДИ РЕПЛИК

Выделение лидера среди реплик. Один из способов масштабирования CRON-задач в распределенных системах – выбор лидера среди реплик. Этот подход заключается в выделении экземпляра ответственного за выполнение чувствительной задачи.

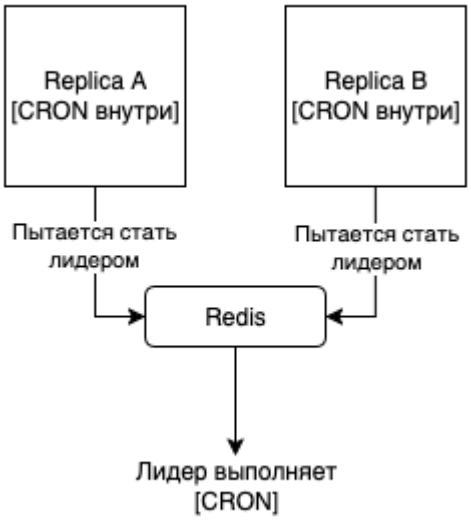

Рис. 2. Схематичное изображение выбора среди реплик

Выбор лидера реализуется с помощью механизма распределенной блокировки. Каждая реплика сервиса пытается записать свой уникальной идентификатор в redis в качестве значения ключа. Тот, кому это удалось сделать первым и становится лидером. Ниже представлен пример кода на JavaScript, реализующий этот подход:

```
async function doDistributedLock()
{
  const lockKey = 'lock:cron:task-
name'
  const lockId = uuid() //
Уникальный идентификатор
(например, uuid.v4())
  const ttl = 30000

  const acquired = await
redis.set(lockKey, lockId, {
```

```
    NX: true, // Только если ключ
не существует
    PX: ttl   // Установим TTL =
30 секунд
  })

  if (acquired) {
    try {
      const currentLockId = await
redis.get(lockKey)
      if (currentLockId ===
lockId) {
        await doCronTask()
      }
    } finally {
      const currentLockId = await
redis.get(lockKey)
      if (currentLockId ===
lockId) {
        await redis.del(lockKey)
      }
    }
  }
}
```

В системах, где необходима большая надежность, используется redlock – вариация распределенной блокировки. Этот алгоритм использует несколько независимых экземпляров redis (от трех до пяти) и требует кворума (подтверждения от большинства реплик), прежде чем считать блокировку успешно захваченной. Пример кода на js с применением библиотеки readlock:

```
import * as Redlock from 'redlock'

const redlock = new
Redlock([redisClient])
const lock = await
redlock.acquire(['locks:cron:task'
], 30000)

try {
  await doCronTask()
} finally {
  await lock.release()
}
```

Выбор лидера среди реплик – надежный способ масштабирования [12] периодических задач, обеспечивающий защиту от дублирующего выполнения. Однако важно учитывать, что этот подход требует тщательной реализации, сложен в тестировании и отладке.

IV. ИСПОЛЬЗОВАНИЕ KUBERNETES CRONJOB

Использование K8S CronJob. Ещё одним способом масштабирования периодических задач в распределенных системах [13] является использование механизма CronJob, встроенного в Kubernetes. В отличие от реализации через бизнес-логику приложения [14], здесь задачи планируются и запускаются самим оркестратором, что позволяет упростить архитектуру и минимизировать дублирование при масштабировании [15].

Суть подхода заключается в том, что по заданному расписанию Kubernetes [16] создаёт под, в котором запускается одна единственная задача. Расписание задается в формате cron-выражения, где указывается точное время запуска. Например, можно настроить выполнение задачи первого числа каждого месяца в полночь. Каждый раз при наступлении нужного времени Kubernetes сам создаёт временный под, запускает внутри него нужный контейнер, а после завершения – удаляет.

Такой подход гарантирует, что вне зависимости от количества реплик приложения задача будет выполнена ровно один раз. Это решает проблему дублирующего исполнения, которая часто возникает при горизонтальном масштабировании сервисов с внутренним cron-расписанием. Преимущество этого способа – автономность. CronJob не зависит от состояния приложения или его реплик. Если даже основное приложение временно недоступно, задача всё равно будет выполнена – отдельно, в изолированном окружении. Кроме того, Kubernetes предоставляет встроенные механизмы управления историей выполнений, ведения логов и повторного запуска в случае неудачи [17]. Однако у данного способа есть и ограничения. Задача должна быть описана в виде отдельного контейнера или команды, которую можно запустить независимо. Также важно помнить, что задержки в планировщике Kubernetes могут повлиять на точность старта задачи, особенно при высокой нагрузке на кластер. Кроме того, при необходимости взаимодействия задачи с другими микросервисами или состоянием системы, может потребоваться дополнительная настройка сетевого доступа и переменных окружения.

Таким образом, использование Kubernetes CronJob может быть эффективным решением для тех случаев, где важна надежность, предсказуемость и изоляция выполнения [18] периодических задач, особенно в системах, уже оркестрируемых в Kubernetes.

V. ПРИМЕНЕНИЕ ДОПОЛНИТЕЛЬНЫХ АТРИБУТОВ В БАЗЕ ДАННЫХ

Другой способ масштабирования периодических задач в распределенных системах – это использование отдельной таблицы блокировок в базе данных. Таблица необходима

для отслеживания состояния выполнения задач в нескольких репликах одного и того же сервиса [19, 20]. В таблице, как правило, присутствует логический атрибут (например, is_running), указывающий, запущена ли задача в текущий момент. Каждая реплика, инициирующая выполнение CRON задачи, сначала прочитывает текущий статус и в случае, если is_running = false, меняет этот статус в true и начинает выполнение задачи [21].

Рис. 3. Пример таблицы «cron_locks»

Этот метод хорош тем, что не требует дополнительной инфраструктуры. Однако, в момент записи значения true в поле is_running возникает состояние гонки данных (race condition). Реплики могут одновременно прочитать is_running = false и, придя к выводу, что задача не запущена, перевести поле в true. В итоге получится двойное выполнение задачи, что, разумеется, нарушает корректность бизнес-логики.

Для того чтобы избежать этой проблемы критически важно использовать транзакционный подход с пессимистической блокировкой. Делаем запрос, который проверяет is_running и изменяет его. Однако, этот запрос оборачивается в транзакцию с конструкцией

```
SELECT … FOR UPDATE
```

которая блокирует сроку для других транзакций:

```
BEGIN;

SELECT is_running
FROM cron_locks
WHERE name = "task_name"
FOR UPDATE;

-- если в поле is_running сейчас
false, то делаем:

UPDATE cron_locks
SET is_running = true
WHERE name = "task_name";

COMMIT;
```

Такая реализация гарантирует, что из всех реплик ответственной за выполнение CRON-задачи будет одна из них.

V. ИСПОЛЬЗОВАНИЕ ОЧЕРЕДЕЙ ЗАДАЧ НА ПРИМЕРЕ BULL

Также для масштабирования периодических задач часто обращаются к очередям задач. Примечательным примером такого инструмента является Bull, который использует Redis для хранения операций [22].

Этот метод позволяет объединить планирование и мониторинг выполнения задач в одном месте [23, 24]. Даже если существует множество реплик одного и того же сервиса, задача выполняется только один раз.

Метод Bull предлагает понятный интерфейс для настройки интервалов выполнения задач с использованием формата CRON. Например, чтобы запустить задание первого числа каждого месяца в полночь, можно использовать следующую конструкцию:

Когда наступит заданное время, задача автоматически добавится в очередь. Затем одна из реплик приложения возьмёт её на выполнение, а остальные реплики проигнорируют эту задачу, поскольку она уже закреплена за конкретным исполнителем.

```
queue.add('job', {}, {
   repeat: {
    cron: '0 0 1 * *'
  }
})
```

ЗАКЛЮЧЕНИЕ

В условиях роста популярности микросервисной архитектуры и горизонтального масштабирования систем вопросы эффективного управления CRON-задачами в распределённых системах становятся особенно актуальными. В статье рассмотрены ключевые проблемы, возникающие при масштабировании таких задач: дублирование выполнения, гонки данных и избыточная нагрузка на внешние узлы. Эти проблемы, проиллюстрированные примерами из продуктовой разработки, такими как еженедельная рассылка писем и обнуление внутренней валюты, подчеркивают необходимость надежных и отказоустойчивых решений.

Анализ пяти способов масштабирования CRON-задач – вынос задач в отдельный сервис, выбор лидера среди реплик, использование Kubernetes CronJob, применение дополнительных атрибутов в базе данных и использование очередей задач на примере Bull – показал их разнообразие и применимость в зависимости от требований системы. Каждый метод имеет свои преимущества: вынос в отдельный сервис упрощает архитектуру, выбор лидера обеспечивает отказоустойчивость, Kubernetes CronJob предлагает изоляцию задач, атрибуты в базе данных минимизируют инфраструктурные затраты, а очереди задач, такие как Bull, обеспечивают гибкость и мониторинг. Однако каждый из них сопряжён с определенными ограничениями, такими как сложность реализации, зависимость от инфраструктуры или необходимость управления транзакциями.

Проведенное исследование подтверждает, что выбор подходящего метода масштабирования зависит от специфики системы, ее масштаба и требований к надежности. Например, для высоконагруженных систем [25] с миллионами задач очереди задач и Kubernetes CronJob могут быть предпочтительнее, тогда как для небольших систем достаточно простых решений, таких как атрибуты в базе данных. Эти выводы, основанные на анализе теоретических подходов и практических примеров, легли в основу разработки решения, обеспечивающего надежное и предсказуемое выполнение CRON-задач в распределенной среде.

БИБЛИОГРАФИЯ

[1] Радостев Д. К., Никитина Е. Ю. Стратегия миграции программного кода из монолитной архитектуры в микросервисы // Вестник Пермского университета. Математика. Механика. Информатика. – 2021. – № 2(53). – С. 65-68. – DOI: 10.17072/1993-0550-2021-2-65-68.

[2] Волков М. Ю., Малецкая М. В., Розов А. С. Сравнение использования монолитной и микросервисной архитектуры веб-приложения в сфере телекоммуникации // Научно-технический вестник Поволжья. – 2024. – № 7. – С. 277-280.

[3] Шевченко В. А., Медведев М. Ю., Назаркин А. С. Декомпозиция сложной динамической системы с сетевой архитектурой на базе диакоптики Крона // Инженерный вестник Дона. – 2018. – № 3(50). – С. 86.

[4] Громей Д. Д., Лебеденко Е. В. Математическое обеспечение поддержки процесса управления схемой реляционной базы данных в задачах горизонтального масштабирования // Моделирование, оптимизация и информационные технологии. – 2019. – Т. 7, № 2(25). – С. 65-79. – DOI: 10.26102/2310-6018/2019.25.2.006.

[5] Схема горизонтального масштабирования веб-сервисов на основе протокола websocket / В. А. Алексеев, П. А. Домашнев, Т. В. Лаврухина, О. А. Назаркин // Системы управления и информационные технологии. – 2018. – № 3(73). – С. 52-55.

[6] Чирков И. А., Дунаев М. Е. Исследование возможностей нейронных сетей для прогнозирования показателей функционирования брокеров сообщений технологических платформ // International Journal of Open Information Technologies. – 2021. – Т. 9, № 8. – С. 36-42.

[7] Разработка унифицированной модели для расчета времени HTTP-запросов на примере веб-приложений Express и fastapi / В. С. Кошельков, Т. А. Грязев, М. А. Соколов, Н. Н. Жуков // Современные наукоемкие технологии. – 2024. – № 5-1. – С. 57-63. – DOI: 10.17513/snt.40005.

[8] Lapkina A. V., Petukhov A. A. HTTP-request classification in automatic web application crawling // Proceedings of the Institute for System Programming of the RAS. – 2021. – Vol. 33, No. 3. – P. 77-86. – DOI: 10.15514/ISPRAS-2021-33(3)-6.

[9] Лапин А. В., Мадумаров Т. А. Использование механизмов удаленного вызова процедур при отладке и тестировании встраиваемых систем космического назначения // Вопросы электромеханики. Труды ВНИИЭМ. – 2023. – Т. 196, № 5. – С. 14-19.

[10] Воронова О. В., Ильин И. В., Харева В. А. Разработка архитектурной модели бизнес-сервисов системы взаимодействия с потребителями сетевых торговых компаний // Известия Санкт-Петербургского государственного экономического университета. – 2020. – № 6(126). – С. 86-92.

[11] Мороз В. В., Мотайленко Л. В. Использование репликации базы данных для горизонтального масштабирования информационных систем // Прикладная математика и информатика: современные исследования в области естественных и технических наук: Материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых, Тольятти, 24–25 апреля 2017 года. – Тольятти: Издатель Качалин Александр Васильевич, 2017. – С. 373-375.

[12] Липатников В. А., Шевченко А. А. Способ контроля уязвимостей при масштабировании автоматизированной системы менеджмента предприятия интегрированной структуры // Информационные системы и технологии. – 2016. – № 2(94). – С. 128-140.

[13] Bagirova S. A., Aliyev A. A. Synchronizing mechanism in distributed information processing systems // Системы управления и информационные технологии. – 2021. – No. 2(84). – P. 44-48. – DOI: 10.36622/VSTU.2021.84.2.010.

[14] Козлов С. В., Седенков С. А. Анализ LSTM и GRU моделей для построения прогнозов временных рядов // International Journal of Open Information Technologies. - 2024. - Т. 12. № 7. - С. 43-50.

[15] Козин А. А., Харченко А. В. Архитектура микросервисов для небольших команд // Прикладная математика: современные проблемы математики, информатики и моделирования: Материалы VI Всероссийской научно-практической конференции, молодых ученых, Краснодар, 15–21 апреля 2024 года. – Краснодар: Краснодарский ЦНТИ – филиал ФГБУ "РЭА" Минэнерго России, 2024. – С. 402-405.

[16] Carrión C. Kubernetes as a Standard Container Orchestrator – A Bibliometric Analysis // Journal of Grid Computing. – 2022. – Vol. 20, No. 4. – P. 42. – DOI: 10.1007/s10723-022-09629-8.

[17] Ding Zh., Wang S., Jiang Ch. Kubernetes-Oriented Microservice Placement with Dynamic Resource Allocation // IEEE Transactions on Cloud Computing. – 2023. – Vol. 11, No. 2. – P. 1777-1793. – DOI: 10.1109/tcc.2022.3161900.

[18] Dell'immagine G., Soldani Ja., Brogi A. KubeHound: Detecting Microservices' Security Smells in Kubernetes Deployments // Future Internet. – 2023. – Vol. 15, No. 7. – P. 228. – DOI: 10.3390/fi15070228.

[19] Данилов А. Д., Синюков Д. С. Подход к управлению транзакциями в гетерогенных распределенных реплицированных системах баз данных в реальном масштабе времени // Системы управления и информационные технологии. – 2021. – № 3(85). – С. 59-65. – DOI: 10.36622/VSTU.2021.85.3.011.

[20] Саенко И. Б., Удальцов А. В., Ермаков А. В. Анализ проблемы синхронизации локальных баз данных в распределенной информационной системе // Труды Научно-исследовательского института радио. – 2022. – № 4. – С. 37-41. – DOI: 10.34832/NIIR.2022.11.4.004.

[21] Лобачев А. Ю., Засов В. А. Коллективный алгоритм обнаружения состояний гонки данных в многопоточных

системах // Вестник СамГУПС. – 2022. – № 4(58). – С. 101-109.

[22] Жиганов А. И., Кожевников П. В., Афонин А. Ю. Разработка программного обеспечения высоконагруженных систем управления IOT устройствами для встраиваемых систем с масштабируемой логикой управления // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник научных статей IV ежегодной межвузовской научно-практической конференции, Пенза, 15 марта 2017 года. – Пенза: Пензенский государственный университет, 2017. – С. 148-150.

[23] Козлов С. В. Использование математического аппарата импликативных матриц при создании и сопровождении информационных систем // International Journal of Open Information Technologies. – 2017. – Т. 5, № 12. – С. 16-23.

[24] Козлов С. В. Математические особенности использования возможностей программного комплекса "Advanced Tester" как инструмента функционального анализа системных данных // International Journal of Open Information Technologies. – 2019. – Т. 7, № 2. – С. 21-30.

[25] Татарникова Т. М., Архипцев Е. Д., Кармановский Н. С. Определение размера кластера и числа реплик высоконагруженных информационных систем // Известия высших учебных заведений. Приборостроение. – 2023. – Т. 66, № 8. – С. 646-651. – DOI: 10.17586/0021-3454-2023-66-8-646-651.

Scaling CRON tasks in distributed systems

S.V. Kozlov, I.A. Karpov

***Abstract* – The article briefly outlines the purpose of CRON tasks as a mechanism for running scheduled tasks. Examples of scheduled product tasks are provided. The need for horizontal scaling of services in a microservice architecture is briefly described. The problems that arise during horizontal CRON scaling in distributed systems are described: depending on the number of replicas repeated or repeated execution of logic, data races leading to anomalies, additional load on external nodes. The problems under consideration are illustrated by cases from product development: the weekly sending of emails and the zeroing of accumulated domestic currency. A description and analysis of five scaling methods is presented: making CRON a separate service, assigning a leader among replicas, using kubernetes CronJob, applying additional attributes to the database, and using task queues using Bull as an example. In the fourth method, additional attributes in the tables act as a flag indicating whether a specific task has started, which avoids duplication. The nuances of implementation, advantages and disadvantages of each method are described. The relevance of the article is due to the widespread use of scheduled tasks in highly loaded distributed systems and the need for their reliable, predictable and fault-tolerant scaling.**

***Keywords* – CRON tasks, scaling, microservices, fault tolerance, data race, software architecture, high-load systems, data consistency.**

REFERENCES

[1] Radostev D. K., Nikitina E. Ju. Strategija migracii programmnogo koda iz monolitnoj arhitektury v mikroservisy // Vestnik Permskogo universiteta. Matematika. Mehanika. Informatika. – 2021. – # 2(53). – S. 65-68. – DOI: 10.17072/1993-0550-2021-2-65-68.

[2] Volkov M. Ju., Maleckaja M. V., Rozov A. S. Sravnenie ispol'zovanija monolitnoj i mikroservisnoj arhitektury veb-prilozhenija v sfere telekommunikacii // Nauchno-tehnicheskij vestnik Povolzh'ja. – 2024. – # 7. – S. 277-280.

[3] Shevchenko V. A., Medvedev M. Ju., Nazarkin A. S. Dekompozicija slozhnoj dinamicheskoj sistemy s setevoj arhitekturoj na baze diakoptiki Krona // Inzhenernyj vestnik Dona. – 2018. – # 3(50). – S. 86.

[4] Gromej D. D., Lebedenko E. V. Matematicheskoe obespechenie podderzhki processa upravlenija shemoj reljacionnoj bazy dannyh v zadachah gorizontal'nogo masshtabirovanija // Modelirovanie, optimizacija i informacionnye tehnologii. – 2019. – T. 7, # 2(25). – S. 65-79. – DOI: 10.26102/2310-6018/2019.25.2.006.

[5] Shema gorizontal'nogo masshtabirovanija veb-servisov na osnove protokola websocket / V. A. Alekseev, P. A. Domashnev, T. V. Lavruhina, O. A. Nazarkin // Sistemy upravlenija i informacionnye tehnologii. – 2018. – # 3(73). – S. 52-55.

[6] Chirkov I. A., Dunaev M. E. Issledovanie vozmozhnostej nejronnyh setej dlja prognozirovanija pokazatelej funkcionirovanija brokerov soobshhenij tehnologicheskih platform // International Journal of Open Information Technologies. – 2021. – T. 9, # 8. – S. 36-42.

[7] Razrabotka unificirovannoj modeli dlja rascheta vremeni HTTP-zaprosov na primere veb-prilozhenij Express i fastapi / V. S. Koshel'kov, T. A. Grjazev, M. A. Sokolov, N. N. Zhukov // Sovremennye naukoemkie tehnologii. – 2024. – # 5-1. – S. 57-63. – DOI: 10.17513/snt.40005.

[8] Lapkina A. V., Petukhov A. A. HTTP-request classification in automatic web application crawling // Proceedings of the Institute for System Programming of the RAS. – 2021. – Vol. 33, No. 3. – P. 77-86. – DOI: 10.15514/ISPRAS-2021-33(3)-6.

[9] Lapin A. V., Madumarov T. A. Ispol'zovanie mehanizmov udalennogo vyzova procedur pri otladke i testirovanii vstraivaemyh sistem kosmicheskogo naznachenija // Voprosy jelektromehaniki. Trudy VNIIJeM. – 2023. – T. 196, # 5. – S. 14-19.

[10] Voronova O. V., Il'in I. V., Hareva V. A. Razrabotka arhitekturnoj modeli biznes-servisov sistemy vzaimodejstvija s potrebiteljami setevyh torgovyh kompanij // Izvestija Sankt-Peterburgskogo gosudarstvennogo jekonomicheskogo universiteta. – 2020. – # 6(126). – S. 86-92.

[11] Moroz V. V., Motajlenko L. V. Ispol'zovanie replikacii bazy dannyh dlja gorizontal'nogo masshtabirovanija informacionnyh sistem // Prikladnaja matematika i informatika: sovremennye issledovanija v oblasti estestvennyh i tehnicheskih nauk: Materialy III nauchno-prakticheskoj vserossijskoj konferencii (shkoly-seminara) molodyh uchenyh, Tol'jatti, 24–25 aprelja 2017 goda. – Tol'jatti: Izdatel' Kachalin Aleksandr Vasil'evich, 2017. – S. 373-375.

[12] Lipatnikov V. A., Shevchenko A. A. Sposob kontrolja ujazvimostej pri masshtabirovanii avtomatizirovannoj sistemy menedzhmenta predprijatija integrirovannoj struktury // Informacionnye sistemy i tehnologii. – 2016. – # 2(94). – S. 128-140.

[13] Bagirova S. A., Aliyev A. A. Synchronizing mechanism in distributed information processing systems // Sistemy upravlenija i informacionnye tehnologii. – 2021. – No. 2(84). – P. 44-48. – DOI: 10.36622/VSTU.2021.84.2.010.

[14] Kozlov S. V., Sedenkov S. A. Analiz LSTM i GRU modelej dlja postroenija prognozov vremennyh rjadov // International Journal of Open Information Technologies. - 2024. - T. 12. # 7. - S. 43-50.

[15] Kozin A. A., Harchenko A. V. Arhitektura mikroservisov dlja nebol'shih komand // Prikladnaja matematika: sovremennye problemy matematiki, informatiki i modelirovanija: Materialy VI Vserossijskoj nauchno-prakticheskoj konferencii, molodyh uchenyh, Krasnodar, 15–21 aprelja 2024 goda. – Krasnodar: Krasnodarskij CNTI – filial FGBU "RJeA" Minjenergo Rossii, 2024. – S. 402-405.

[16] Carrión C. Kubernetes as a Standard Container Orchestrator – A Bibliometric Analysis // Journal of Grid Computing. – 2022. – Vol. 20, No. 4. – P. 42. – DOI: 10.1007/s10723-022-09629-8.

[17] Ding Zh., Wang S., Jiang Ch. Kubernetes-Oriented Microservice Placement with Dynamic Resource Allocation // IEEE Transactions on Cloud Computing. – 2023. – Vol. 11, No. 2. – P. 1777-1793. – DOI: 10.1109/tcc.2022.3161900.

[18] Dell'immagine G., Soldani Ja., Brogi A. KubeHound: Detecting Microservices' Security Smells in Kubernetes Deployments // Future Internet. – 2023. – Vol. 15, No. 7. – P. 228. – DOI: 10.3390/fi15070228.

[19] Danilov A. D., Sinjukov D. S. Podhod k upravleniju tranzakcijami v geterogennyh raspredelennyh replicirovannyh sistemah baz dannyh v real'nom masshtabe vremeni // Sistemy upravlenija i informacionnye tehnologii. – 2021. – # 3(85). – S. 59-65. – DOI: 10.36622/VSTU.2021.85.3.011.

[20] Saenko I. B., Udal'cov A. V., Ermakov A. V. Analiz problemy sinhronizacii lokal'nyh baz dannyh v raspredelennoj

informacionnoj sisteme // Trudy Nauchno-issledovatel'skogo instituta radio. – 2022. – # 4. – S. 37-41. – DOI: 10.34832/NIIR.2022.11.4.004.
[21] Lobachev A. Ju., Zasov V. A. Kollektivnyj algoritm obnaruzhenija sostojanij gonki dannyh v mnogopotochnyh sistemah // Vestnik SamGUPS. – 2022. – # 4(58). – S. 101-109.
[22] Zhiganov A. I., Kozhevnikov P. V., Afonin A. Ju. Razrabotka programmnogo obespechenija vysokonagruzhennyh sistem upravlenija IOT ustrojstvami dlja vstraivaemyh sistem s masshtabiruemoj logikoj upravlenija // Informacionnye tehnologii v nauke i obrazovanii. Problemy i perspektivy: sbornik nauchnyh statej IV ezhegodnoj mezhvuzovskoj nauchno-prakticheskoj konferencii, Penza, 15 marta 2017 goda. – Penza: Penzenskij gosudarstvennyj universitet, 2017. – S. 148-150.
[23] Kozlov S. V. Ispol'zovanie matematicheskogo apparata implikativnyh matric pri sozdanii i soprovozhdenii informacionnyh sistem // International Journal of Open Information Technologies. – 2017. – T. 5, # 12. – S. 16-23.
[24] Kozlov S. V. Matematicheskie osobennosti ispol'zovanija vozmozhnostej programmnogo kompleksa "Advanced Tester" kak instrumenta funkcional'nogo analiza sistemnyh dannyh // International Journal of Open Information Technologies. – 2019. – T. 7, # 2. – S. 21-30.
[25] Tatarnikova T. M., Arhipcev E. D., Karmanovskij N. S. Opredelenie razmera klastera i chisla replik vysokonagruzhennyh informacionnyh sistem // Izvestija vysshih uchebnyh zavedenij. Priborostroenie. – 2023. – T. 66, # 8. – S. 646-651. – DOI: 10.17586/0021-3454-2023-66-8-646-651.